ID-based cryptography using symmetric primitives

Chris J. Mitchell, Fred C. Piper and Peter R. Wild
Information Security Group, Royal Holloway,
University of London, Egham, Surrey TW20 0EX, UK
{c.mitchell,f.piper,p.wild}@rhul.ac.uk

23rd May 2007

Abstract

A general method for deriving an identity-based public key cryptosystem from a one-way function is described. We construct both ID-based signature schemes and ID-based encryption schemes. We use a general technique which is applied to multi-signature versions of the one-time signature scheme of Lamport and to a public key encryption scheme based on a symmetric block cipher which we present. We make use of one-way functions and block designs with properties related to cover-free families to optimise the efficiency of our schemes.

Keywords: ID-based cryptography, symmetric primitives, one-time signature, cover-free family.

AMS classification: 94A60

1 Introduction

According to Menezes, van Oorschot and Vanstone [17], identity-based (ID-based) cryptography was first proposed by Blom [2] in 1983. Since then a large variety of different schemes have been proposed, a summary of which (up to 1997) can be found in [17]. ID-based signature schemes have probably received the most attention, and an international standard exists for such schemes, [11].

As defined in [17], an ID-based scheme is an asymmetric cryptosystem wherein the entity's public identification information plays the role of its public key, and is used by a trusted authority T (along with T's private key) to compute the entity's private key. After computing it, T transfers the entity's private key to the entity over a secure channel. In this paper we

describe a general method for deriving an ID-based system from any 'conventional' public key cryptosystem. We describe an application of this idea to derive an ID-based key establishment scheme in the setting of public key cryptosystems based on the discrete logarithm problem.

Our method is also applicable to the less conventional public key cryptosystems which are based on symmetric primitives. We adopt the one-time signature schemes that use symmetric primitives to develop an ID-based signature scheme. We also present a public key encryption scheme using a symmetric block cipher and develop an ID-based scheme from it.

Note that, in the schemes derived using the general method described below, we use a slightly generalised version of the above definition of ID-based scheme. The entity's public key is actually derived from the entity's identification information using a public function (this is, in fact, a common property of such schemes).

As discussed below, of the possible types of ID-based cryptosystem, encryption and key establishment schemes are likely to be of the greatest practical significance. They also happen to be amongst the least common, which makes the work of this paper of potential practical importance. Such schemes enable a user to send a secret message to any other entity, without having to first obtain any key material for that entity. All that is required is an identifier for the recipient, e.g. an email address. It only requires the sender to first acquire the appropriate set of system parameters. Moreover, the recipient does not need to have been equipped with the private decryption key in advance — he/she can request it when needed. This type of scheme is probably most appropriate when the recipient is part of a well-defined organisation, e.g. a government department or a large corporation, for which system parameters can be made widely available. There are many examples of situations where the need to send a secure message to such an individual arises, not least for e-government.

In section 2 we give our technique for deriving an ID-based scheme from any public key cryptosystem and section 3 discuss a specific implementation and some related practical issues. In section 4 we discuss public key cryptosystems based on symmetric primitives. We review signature schemes based on the one-time signature scheme of Lamport and describe a multi-signature version of Rabin's one-time signature scheme. We show how a public key encryption scheme may be derived from a symmetric block cipher system and discuss the use of Merkle hash-trees for efficient authentication of public keys and verification data. In section 5 we show how to obtain ID-based signatures from symmetric primitives and how to optimize their efficiency and in section 6 we show how to obtain an ID-based encryption scheme from symmetric primitives.

2 Deriving an ID-based cryptosystem from any public key cryptosystem

We start by describing a method which enables any public key cryptosystem to be used as an ID-based scheme. In the description below we suppose that a trusted centre has been chosen. As for any ID-based scheme, this centre must be trusted by all the users of the scheme. We also suppose that a public key cryptosystem has been selected for use — this can be a public key cryptosystem of any type.

Note that the scheme described here is, as far as is known, original. However, there are some resemblances to the key predistribution scheme of Matsumoto and Imai [15], although the latter scheme is designed only for establishing shared secret keys.

Heng and Kurosawa [8] describe a k-resilient identity-based encryption scheme that makes use of the homomorphic properties of El Gamal encryption. Their scheme is secure under the assumption that the Decisional Diffie-Hellman problem is hard, provided that no more than k private keys are subverted. The scheme of Boneh and Franklin [3], which uses the bilinearity property of the Weil pairing, is secure under the Weil Diffie-Hellman assumption.

Our technique is combinatorial in nature and does not rely on any algebraic properties of the public key cryptosystem. Indeed, we are able to make use of asymmetric cryptosystems based on symmetric primitives. In such a case, the private key is typically a string of keys for a block cipher. Although the security level of our scheme for an adversary without access to these keys is that of the hash function or block cipher used in the construction, an adversary with access to subverted keys may succeed in an attack with probability the ratio of the number of subverted keys to the total number of keys held by a user.

2.1 Initialisation

The trusted centre T first generates a large number of public/private key pairs for the chosen public key cryptosystem. The set of public keys are then distributed to all parties in the system, whilst the private keys are retained by T (these keys collectively form T's private key for the ID-based cryptosystem). We suppose that a total of n public keys are distributed, and that they are labelled $P_0, P_1, \ldots, P_{n-1}$. Note that this distribution must be performed in such a way that the recipient can trust the validity of the information. This is again a common feature of ID-based schemes; almost invariably it is necessary for T to distribute domain parameters in a trusted way. The main difference here is that the amount of information

is significantly larger than usual (although not too large to cause major problems, as we discuss below).

T also selects and distributes a 'key finding' function f, where f maps I, the set of all possible user identifiers (within the community of users served by T), into the set $\{0, 1, \ldots, n-1\}$. The choice of f will depend on the application; however, in general f should be chosen to minimise the probability that $f(i_X) = f(i_Y)$ when $i_X \neq i_Y$ $(i_X, i_Y \in I)$. Of course if $|I| \leq n$ then f can be chosen to be injective, i.e. this probability is zero. However, the more typical case is likely to be where $|I|$ is very much larger than n, and in this case one can implement f using a hash-function with good randomising properties (i.e. so that each output of f is equally likely). Note that we only require the properties normally associated with a hash-function as used for data storage, rather than requiring all the properties normally required of a cryptographic hash-function. (Even if f is realised using a one-way hash-function, f will not be one-way since n will not be sufficiently large for this to be possible).

2.2 Obtaining keys

Now suppose that entity A wishes to obtain the public key for entity B. We also suppose that entity A knows the identifier i_B of B, where this identifier must be unique within the particular community of users. To obtain the desired public key, A computes $f(i_B)$, and the public key of B is simply $P_{f(i_B)}$. Hence A can obtain B's public key purely using information already possessed by A.

When B wishes to obtain his/her private key, he/she needs to obtain it from T. Assuming that the n private keys held by T are labelled $S_0, S_1, \ldots, S_{n-1}$, where S_j is the private key corresponding to public key P_j $(0 \leq j < n)$, T provides B with the private key $S_{f(i_B)}$. Of course, before handing over the key, T would typically check that i_B is the valid identifier for B, and that B is entitled to be given the private key. (This will prevent B from choosing his/her identifier so that $f(i_B) = f(i_C)$ for another user C).

Note that one obvious alternative to our scheme would simply be to allocate keys to entities prior to distribution of the set of public keys. However, the advantage of our scheme is that it allows entities to be equipped with key pairs at any time, without redistributing public keys.

2.3 Properties and parameter choice

The system will clearly work, and the main problem would appear to be that there is a chance that two different entities (A and B say) will be assigned the same key pair. Assuming that $P_0, P_1, \ldots, P_{n-1}$ are all distinct, this will

happen if and only if $f(i_A) = f(i_B)$. The probability that this will happen depends on the choice of f, the number of users (u say), and the value of n.

For example, if f is implemented using a hash-function, and we assume that the hash-function behaves randomly, then the probability that this will happen is equal to

$$1 - \prod_{j=0}^{u-1} \frac{n-j}{n}.$$

For large n this approximates to $1 - \exp(-u^2/2n)$; see, for example, [17]. Hence, by choosing the values so that u is a little less than $\sqrt{n}$, the probability of this undesirable occurrence can be made suitably small.

Of course, this discussion reveals a fundamental limitation of our approach by comparison with other schemes, such as those of Boneh-Franklin [3] and Cocks [5]. That is, in these latter schemes identifiers can be chosen arbitrarily, whereas here the form of identifiers must be constrained lest an attacker is able to generate unlimited numbers of identifiers which can be legitimately associated with the attacker. If this was possible then the attacker could repeatedly generate identifiers for itself until one is found which has the same image under f as the identifier of a target user.

3 An example implementation

We next consider one example of the derivation method described in the previous section. We set this example in the more familiar setting of public key cryptosystems that depend on the discrete logarithm problem. This is the setting of the archetypical Diffie-Hellman key agreement protocol. We consider ID-based cryptosystems that depend on symmetric primitives in sections 4, 5 and 6.

3.1 Background

We suppose that the public key cryptosystem to which the derivation technique applies is one of the variants of the Diffie-Hellman key agreement scheme, as originally proposed in [7]; for a discussion of some of the many variants of this scheme, see, for example, [17], or the relevant international standard, [10]. With all these schemes, a large prime p is chosen together with a 'base' g, where g should be chosen to have large prime order modulo p (let q be the order of g mod p). A private key is then a random integer x less than q, and the corresponding public key is simply g^x mod p.

One issue of significance here is that this can yield a non-interactive key agreement scheme, where the entity wishing to derive a shared secret can do

so using the scheme without any prior exchange of messages. Such schemes appear relatively difficult to construct — see, for example, [16]. More generally, the scheme could also be used to yield an ID-based encryption scheme, examples of which are also not common (see [3, 6]).

Note also that, in one sense at least, non-interactive ID-based key agreement schemes are potentially amongst the most useful of the ID-based schemes, since they enable one user to send an encrypted message to another user with no prior interactions with that user or with a trusted third party (except the initial interaction to establish the ID-based private key). This is not possible with a non-ID-based key agreement scheme. ID-based signatures are a little less attractive in this respect, since even with a conventional signature scheme it is possible to send a signed message with no prior interactions, and indeed arrange for the recipient to be able to verify it immediately, simply by attaching the appropriate public key certificate to the signed message.

Finally note that the discussion below could apply equally to any public key cryptosystem in which a user's public key is the discrete logarithm of his/her private key in some group (e.g. the elliptic curve group).

3.2 Realisation

The trusted centre generates g and p, and also generates n key pairs. The set of n public keys (together with g and p) then need to be distributed to all members of the scheme. One possibility would be to write all n public keys to a portable storage medium, e.g. a DVD-ROM. A double sided DVD-ROM has the potential to store of the order of 2^{33} bytes of data (i.e. around 8 Gigabytes). This DVD-ROM could then be distributed in a secure way to all participants in the scheme, and the data could then (if necessary) be copied to the hard disk of each user's PC.

If we assume that the Diffie-Hellman prime p is 1024 bits long (i.e. 2^7 bytes), then the DVD-ROM has the capability of storing of the order of 2^{26} public keys, i.e. around 70 million. Of course, it will take the trusted centre a little while to generate this number of public keys, but generating Diffie-Hellman key pairs is a relatively simple matter, requiring one random number and one multi-precision exponentiation per key pair. With appropriate 'hardware assists' it should be possible to generate 100 such key pairs per second. At such a rate, generating 70 million key pairs would take around 1 week.

It remains to decide how many users one DVD-ROM of this type would safely support, i.e. how many users can be supported without there being too high a probability that two users will share the same key pair. Using the formula above, we have $n = 2^{26}$ and $\sqrt{n} = 2^{13}$, and thus we need to choose u so that $\exp(-u^2/2^{27})$ is close to 1. The value can be chosen using Table 3.2; from this table it should be clear that, for at most 1000 users, the

probability of two users sharing the same key is under 1%.

Table 1: **Probability that all users have distinct keys**

Number of users (u)	Probability keys all distinct
2^{12}	0.88
2^{11}	0.97
2^{10}	0.992

Of course, there are various strategies that could be used to safely expand the number of users sharing a single DVD-ROM. For example, suppose that identities are based on email addresses, and that the trusted centre is responsible for allocating email addresses to new users. When allocating a new email address i_A, the trusted centre could first verify that $f(i_A)$ is distinct from all previously allocated values; if there is a clash the trusted centre simply allocates a slightly different email address.

Before proceeding we observe that schemes with similarities to the above idea, specific to discrete logarithm based schemes, have been proposed by Tsujii and Itoh [26] and Lee and Liao [14]. Note also that the scheme of Lee and Liao possesses significant defects (see, for example, [25]).

4 Public key cryptosystems from symmetric primitives

A number of techniques have been proposed to derive public key cryptosystems from symmetric cryptosystems. We next review some of the more significant of these. The interest in the resulting schemes stems mainly from the advantages of such schemes in terms of speed of operation. However, they do have the disadvantage that the size of the keys is quite large.

Any of these schemes can be combined with the technique described in section 2 to yield an ID-based scheme based purely on symmetric primitives. However, we describe below how, in certain cases, such combinations can be made more efficient.

4.1 Signature schemes

A significant number of signature schemes based purely on one-way functions (or other symmetric crypto-primitives) have been proposed. However, whilst the number of schemes is relatively large, they all appear to be derived from two main ideas, namely the schemes of Lamport [13] and Rabin [23].

These fundamental schemes are all examples of what have become known as *One-time signatures* (OTSs). That is, a private key can only be used

to generate one signature. However, as we briefly outline below, a number of multi-time signature schemes have been designed based on the original Lamport scheme. We first review the Lamport scheme [13].

The idea behind this one-time signature scheme is that secret key material is revealed dependent on the message to be signed. The signature is verified by checking that the key material corresponds to authenticated public verification data of the signer.

In the Lamport scheme, to sign a message of n bits, $m = m_1 m_2 \dots m_n$, the signer chooses $2n$ keys, $K_{1,0}, K_{1,1}, K_{2,0}, K_{2,1}, \dots, K_{n,0}, K_{n,1}$ and publishes the verification data $h(K_{1,0}), h(K_{1,1}), h(K_{2,0}), h(K_{2,1}), \dots, h(K_{n,0}), h(K_{n,1})$ where h is a one-way hash function. The signature on m is the sequence of n keys $K_{1,m_1}, K_{2,m_2}, \dots, K_{n,m_n}$. To forge a signature on a message distinct from m would require the ability to determine a pre-image under h of some element of the public key.

Many schemes based on this idea have been proposed (see, for example, Merkle [19], Bos and Chaum [4], Bleichenbacher and Maurer [1]). The private key is an ordered set $\mathcal{K}$ of random values and the public key is the corresponding ordered set $h(\mathcal{K}) = \{h(K) | K \in \mathcal{K}\}$ for some one-way hash function h. The schemes vary according to the size of the public and private keys, the size of the signature, and the computation required for signature generation and verification. A message determines a subset of K (as in the scheme of Merkle given in [19], or Bos-Chaum) or a partial computation of some subset of $h(\mathcal{K})$ (as in the scheme of Winternitz described in [19] or Bleichenbacher-Maurer) and the signature consists of these values. The signature is verified by checking that the revealed values correspond under h to the appropriate verification values in $h(\mathcal{K})$.

These schemes are secure against forgery because, without knowledge of the private key, the ability to determine a signature on a message (even having observed the signature on a different message) implies the ability to invert a one-way function. They are one-time signatures as the signatures on two distinct messages may allow the generation of a signature on a third message.

Perrig [21], Reyzin and Reyzin [24] and Pieprzyk *et al.* [22] have described extensions of this one-time signature technique to multiple-time signature schemes. In these schemes the key material revealed in some limited number, c say, of signatures cannot be used to forge a signature on another message. This is because the determination (with non-negligible probability) of a message whose signature may be inferred from the subsets of keys that are revealed in c signatures implies the subversion of a one-way function ([21, 24]) or because the subsets of keys that are revealed in c signatures satisfy a combinatorial property so that the union of c such subsets does not contain any other subset ([22]).

We describe the Reyzin and Reyzin scheme for parameters of interest to us. The signer's private key consists of $n = 2^{25}$ keys $K_1, K_2, \ldots, K_n$ of 80 bits each. The public key is $V_1 = h(K_1), V_2 = h(K_2), \ldots, V_n = h(K_n)$ where h is a one-way hash function producing a hash of 160 bits. We select the system parameter $t = 32$. To sign a message m, the hash $H = h(m)$ is used to seed a pseudo-random number generator to produce $t \log_2 n = 800$ bits: $H_0, H_1, \ldots, H_{799}$. These are used to determine $t(= 32)$ integers $i_j = \sum_{\ell=0}^{24} H_{25j+l} 2^\ell$, $j = 0, 1, \ldots, 31$. The signature on message m consists of the keys $K_{i_0}, \ldots, K_{i_{31}}$. A signature $K'_0, \ldots, K'_{31}$ on message m is accepted if and only if $h(K'_j) = V_{i_j}$ for $j = 0, 1, \ldots, 31$. Although the signature is quite large, signature generation and verification are both very quick.

For $n = 2^{25}$ and 160-bit hashes, the public domain parameters will occupy 20×2^{25} bytes, which will fit on a single CD-ROM. The probability that a subset of t keys chosen randomly from a set of n keys lies with a given set of r keys is bounded above by $(\frac{r}{n})^t$. Thus, if $c = 2^{19}$ signatures are observed so that $r = ct = 2^{19}.2^5$ keys are known, then the probability that a signature can be forged is less than $(\frac{ct}{n})^t = (\frac{2^{19}.2^5}{2^{25}})^{32} = 2^{-32}$. Thus the key pair can be used for over half a million signatures.

Merkle [19] has a different approach to extending one-time signatures to multi-signatures. His idea is to use the signature to also authenticate additional key material which can then be used to sign the next message. An efficient way to do this is to use an authentication tree. Authentication trees may also be used to efficiently authenticate the large volume of verification data that one-time signatures generally have. This is discussed below in section 4.3.

4.2 Public key encryption

Although there has been significant interest in developing signature schemes using symmetric primitives, very little attention has been given to designing public key encryption schemes from symmetric primitives. The puzzle system of Merkle [18] was one of the earliest suggested methods of key agreement. We describe this below and then present a public key encryption scheme based on a symmetric block cipher (Mitchell [20]) that uses a related technique similar to that first conceived by Merkle (Weber [27]).

Merkle's puzzle system is based on the existence of a collection of puzzles each of which requires a specified amount of effort to solve. We quantify this as the number n of steps the solver must perform to find the solution to the puzzle, where a step is the effort required to create a single puzzle. For example, given a symmetric block cipher with n possible keys, a puzzle might be to find the key which was used to produce a cryptogram (from a specified plaintext or family of plaintexts). One encryption is used to create

this puzzle; n encryptions of an exhaustive search solves it.

If user A wishes to establish a key with user B, then A creates n puzzles and sends them to B, where each puzzle requires $O(n)$ work to solve. User B then chooses one of the puzzles at random and solves it. Both user A and user B expend $O(n)$ steps of effort, A to create the n puzzles, and B to solve one of them. User A and user B now have common knowledge about this puzzle, and may be able to use it as a shared key with an encryption function. For example, A could choose n random keys from a key space of size n, and use them to encrypt a series of strings of the form $(c||x||f(x))$, where c is a fixed value (known to B), x is randomly chosen for each puzzle, and f is a function known only to A; f could, for example, be implemented by encrypting x with a fixed secret key known only to A. User B then performs an exhaustive search to determine the key corresponding to one of the ciphertexts; the constant c will enable B to know when the correct key has been found. B now uses the value x from the decrypted puzzle as the key index, and the value $f(x)$ as the secret key. B sends x to A, who can compute $f(x)$, and hence A and B share a secret key. Another user would have to solve n puzzles, thereby expending n^2 steps of effort, to discover this key.

We next describe a slightly modified version of the Merkle puzzle scheme, which, like the Merkle scheme, is based on a simple trade-off between storage and computation. An important aspect of all such schemes is that, as the availability of cheap mass-storage grows, so the security of the scheme can be increased. The advantage that this scheme has over Merkle's scheme is that the public key can be made a little smaller (typically around half the size).

The scheme uses an n-bit block cipher, i.e. a block cipher operating on plaintext and ciphertext blocks of n bits for some n. We write $e_K(X)$ for the block cipher encryption of data string X using a key K, which we assume has k bits. We write $d_K(X)$ for the corresponding decryption. Note that if X is longer than the block cipher block length n then, in computing $e_K(X)$, we assume that an appropriate mode of operation (e.g. Cipher Block Chaining — see, for example, [17]) is used. There are two other parameters: m helps determine the size of the public and private keys; and d helps determine the size of public keys (details below).

Key generation is as follows. A user wishing to generate a private/public key pair must obtain a set of m random keys for the specified block cipher, $K_1, K_2, \ldots, K_m$ say. These constitute the user's private key. The user must also choose a random data string X of dn bits, and then compute $P_i = e_{K_i}(X)$ for every i $(1 \leq i \leq m)$. The sequence of values $(P_1, P_2, \ldots, P_m)$ together with X then constitutes the user's public key. Note that every user should choose a different value for X (e.g. by incorporating a unique

identifier into X).

Suppose user A wishes to send user B an encrypted message, and A knows that B's public key is $((P_1, P_2, \ldots, P_m),\ X)$. A now generates a sequence of random block cipher keys: $L_1, L_2, \ldots$, and for each such key generates $e_{L_i}(X)$ and compares the result with $(P_1, P_2, \ldots, P_m)$. Eventually a match will be found, i.e. suppose $e_{L_j}(X) = P_i$, for some i and j. Given that d was chosen to be sufficiently large, there is a very high probability that $L_j = K_i$.

User A now stops the search and retains L_j and i. This constitutes a key shared with B and can be used to encrypt a message for B (as many times as is required). Note that if A retains L_j and i, A can discard the remainder of B's public key.

To encrypt a message M for transmission to B, A simply computes $e_{L_j}(M)$, and sends the result to B, along with the integer i. On receipt of an encrypted string C, together with the parameter i, B uses key K_i to decrypt it, i.e. B computes $d_{K_i}(C)$.

The parameter d should be chosen so that the probability that two keys will map the dn-bit string X to the same encrypted string is very small. Assuming that the block cipher behaves in a random fashion, this can be achieved by choosing dn to be a few bits longer then k.

The parameter m should be chosen to be as large as is feasible. The larger m is, the more efficient the rest of the scheme is and/or the greater the security level can be.

As for the scheme of Heng and Kurosawa [8], in which a (generalised) Diffie-Hellman key exchange is performed to effect a (generalised) El Gamal encryption, this scheme performs a key exchange to effect a block cipher encryption. Heng and Kurosawa establish the security of their scheme under the assumption that the Decisional Diffie-Hellman problem is hard, provided that no more than k private keys are subverted. In our case we assume that there is no more efficient way to attack the block cipher than to search for the keys by creating tables of corresponding plaintext-ciphertext pairs and comparing them with known pairs. Then, since the key used for encryption is chosen by a random process, the probability that the adversary is successful in decrypting a ciphertext is the ratio of the number of keys subverted by this endeavour and m. For example, if $m = 2^{\frac{k}{2}}$, then the expected number of subverted keys is 1, and the probability that the adversary is successful is $2^{-\frac{k}{2}}$.

Finally note that, as we have just discussed, this scheme requires that k has the property that $2^k/m$ block cipher operations is feasible for a legitimate user. That is, implementation of the scheme requires the use of a block cipher with a configurable key length. Under reasonable assumptions about the security of the block cipher, this could reasonably be achieved by taking

a 128-bit block cipher such as AES [12], and converting a k-bit key to a 128-bit key. This conversion could, for example, be achieved by appending a fixed $(128 - k)$-bit string, or applying a 128-bit hash function, to every k-bit key.

4.3 Using Merkle hash-trees

The signature schemes and encryption scheme described above have public keys which may consist of a large number of verification values, amounting to a large amount of data. Moreover for the one-time signature schemes a new public key is required for each signature. These public keys must be authenticated by a trusted authority. A Merkle hash-tree may be used to authenticate a large number of verification values or public keys without the need for each of them to be signed by the trusted authority.

We mentioned in section 4.1 above Merkle's one-time signature scheme, that allows arbitrarily many signatures to be produced by authenticating additional key material with each signature. The hash-tree method we consider here allows some fixed number, say 2^s for convenience, of values to be authenticated in advance of producing signatures.

Let $\mathcal{T}$ be the complete binary tree with 2^s leaves. We may identify the vertices of this tree with the $2^{s+1} - 1$ binary strings of length at most s. The leaves correspond to the strings of length exactly s and the root of the tree corresponds to the empty string, $\emptyset$ of length 0. Two vertices are joined by an edge if the lengths of their corresponding strings differ by 1 and one string is a prefix of the other. Suppose that we wish to authenticate 2^s verification values $V_j = h(K_j)$, $j = 1, \ldots, 2^s$. We associate each value with a leaf of the Merkle hash tree by an appropriate relabelling so that the values are denoted $V_{\mathbf{s}}$ where $\mathbf{s}$ varies over the 2^s leaves (equivalently, $\mathbf{s}$ ranges over the 2^s bitstrings of length s). We recursively define $V_{\mathbf{u}}$ for any string u of length $\ell < s$ by putting $V_{\mathbf{u}} = h(V_{\mathbf{u0}}||V_{\mathbf{u1}})$ where $\mathbf{u0}, \mathbf{u1}$ are the two strings of length $\ell + 1$ with prefix $\mathbf{u}$, $||$ denotes concatenation, and h is a hash function.

Now the trusted authority need only sign $V_\emptyset$ rather than the 2^s values V_j. A user authenticates $V_{\mathbf{s}}$ by providing $V_{\mathbf{s}}$ and $V_{\mathbf{u}}$ for the s vertices $\mathbf{u}$ joined to a vertex on the path from $\mathbf{s}$ to the root $\emptyset$ but not on this path — that is, the very values that allow the recursive evaluation of hash values along the path to obtain $V_\emptyset$. So, in a scheme based on the Reyzin-Reyzin scheme with parameters as described above, only 20 bytes ($V_\emptyset$) are signed by the trusted centre. To enable verification of a signature, along with the the 320 bytes of the signature, $K_{i_0}, \ldots, K_{i_{31}}$, the signer provides $\log_2 n = 25$ hash values $V_{\mathbf{u}}$ (as determined by the hash tree) of 20 bytes each for each of the $t = 32$ verification values $V_{i_0}, \ldots, V_{i_{31}}$. This is an increased communication cost, but provides a significant saving on the amount of material that the

trusted centre must sign.

5 ID-based signatures from symmetric primitives

In section 2 we described how to derive an ID-based cryptosystem from any public key cryptosystem. Therefore, we can obtain an ID-based signature scheme from any of the signature schemes based on symmetric primitives described above. In this naive approach the trusted centre T generates n public/private key pairs, where a private key S_i consists of a set $\mathcal{K}_i$ of secret keys and $P_i = h(\mathcal{K}_i)$. The public keys P_i may be authenticated using hash-trees. A user A with identifier i_A has public key $P_{f(i_A)}$.

In this naive approach we require T to sign a hash-tree root value $V_\emptyset$ for each user. If the number of users is large then there may be efficiency gains by using a hash-tree to authenticate these. A signer would then provide a verifier with values in this hash-tree in order to authenticate the signer's root value.

For the Reyzin-Reyzin scheme described above, T would generate 2^{25} private keys and create a hash-tree for the corresponding verification values for each user.

5.1 An optimised approach

In the naive approach the trusted centre sets up a large number of collections of secret keys and verification values. The ID of a user determines which is their private/public key pair. In this section we describe a scheme in which a private/public key pair is obtained from a pool of secret key/verification value pairs. When the number of users is large this may provide significant savings on generation and storage costs.

The trusted authority T creates a large collection of (secret key, verification value) pairs, where a secret key is a random bit-string K_i, e.g. of 80 bits, and the corresponding verification value is $h(K_i)$ for some one-way hash-function h. The authority T then publishes all the values $h(K_i)$, $1 \leq i \leq N$ (for some large N) — these form the domain parameters for the ID-based scheme. All users of the scheme must have a trusted copy of these parameters. They may be signed by the authority using a hash-tree, for example.

We also need a one-way function f, which, given a bit string as input, gives as output a n-subset of $\{1, 2, \ldots, N\}$, where n is a system parameter ($1 < n < N$). For example, if $N = 2^s$, we could let the j^{th} element of the subset be the integer whose binary representation is given by the first s bits of the output of f^j (with duplicates eliminated, as necessary, and generating additional s-bit values to replace them). This function f forms

the key-finding function of section 2.1. Suppose user U has identifier i_U. Then the public key for user U consists of the collection of n values $h(K_j)$ for which $j \in f(i_U)$. The private key for user U consists of the corresponding collection of values K_j.

Each such private/public key pair can be used with either the Lamport scheme or some multi-use variant of it.

As in our description of the Reyzin-Reyzin scheme, a user's signature may be forged with non-negligible probability if sufficiently many of the user's secret keys are known by another entity. In this scheme the main source of information about a user's keys is not the signatures that the user has produced but the keys held by other users. A coalition of c users could know as many as cn of the N secret keys. The probability that a random choice of t of the user's keys are among these is bounded by $(\frac{cn}{N})^t$. So, with $n = 2^{25}$ and $t = 32$, a coalition of 20 users would have a probability of forging of less than 2^{-32}, if $N = 2^{31}$. This is just 64 times the number of keys required for one user, and so provides a significant efficiency gain for any large system of users.

5.2 Another optimised approach

The scheme described in the previous section relied on the randomness of the one-way function f (as well as the vigilance of the trusted centre when distributing keys, as discussed in section 2.2) to ensure that the keys held by a given user are not also held by a small group of other users. In this section we describe how this may be ensured by design rather than by relying on chance.

The scheme is set up by the trusted authority in the same way as the previous section, but instead of using a randomising one-way function f to determine which keys are given to a user, the trusted authority uses a mapping f into the blocks of a suitable block design. The trusted authority chooses a block design with u blocks and N points. Each block is a subset of n of the points. The points of the design are associated with the (secret key, verification value) pairs. The trusted centre assigns a user with identifier i_U the pairs corresponding to the points of the block $f(i_U)$.

As before, we are concerned that a coalition of c users should not know too many of the keys held by another user. In terms of the design, this means that for any $c+1$ blocks, $B_0, B_1, \ldots, B_c$ of the design, the cardinality of the set

$$B_0 \setminus \cup_{i=1}^{c} B_i$$

should be sufficiently large.

Pieprzyk *et al.* [22] have described how to use cover-free families to make

the Reyzin-Reyzin scheme independent of the existence of one-way functions. The designs we are interested in are cover-free families, and are used in a similar way here. To be cover-free it is sufficient that the above cardinality is always at least 1, so, in general, we require a stronger condition on the collection of blocks.

If sufficiently large is taken to mean that

$$|B_0 \setminus \cup_{i=1}^{c} B_i| \geq \frac{n}{2},$$

then the probability that a coalition of size c may forge a signature is at most 2^{-t}, as in the example of the Reyzin-Reyzin scheme considered above.

We may ensure that this cardinality is sufficiently large by ensuring that the cardinality of the intersection of any two blocks is sufficiently small. Of course, if the blocks are mutually disjoint then we simply have the naive approach described above. The use of a design with non-trivial intersections requires fewer (secret key, verification value) pairs than the naive approach. The following geometric construction gives designs with the appropriate properties.

Consider an elliptic quadric in the projective geometry $\mathrm{PG}(3, q)$ of dimension 3 over the field $\mathrm{GF}(q)$ of q elements. This is described by a quadratic form in the four projective coordinates that has exactly $q^2 + 1$ solution points. This set of $q^2 + 1$ points has the property that no three of them are collinear (see Hirschfeld [9]). Dualising, the corresponding object is a set of $q^2 + 1$ planes of $\mathrm{PG}(3, q)$ with the property that any two intersect in a line and any three have a single point in common. Thus this configuration determines a design with $N = q^3 + q^2 + q + 1$ points and $q^2 + 1$ blocks (the planes) of $q^2 + q + 1$ points each such that any two blocks intersect in $q + 1$ points. Such a design could be used in a scheme where there may be as many as $\frac{q+1}{2}$ users in a coalition. This scheme has $N = q^3 + q^2 + q + 1$ whereas the naive scheme would have $N = q^4 + q^3 + 2q^2 + q + 1$.

For example, with $q = 2^{12}$, discarding one of the planes (nominating it to be the plane at infinity of the projective space) and restricting ourselves to the $N = q^3 = 2^{36}$ affine points, we obtain a scheme for $u = 2^{24}$ users with $n = 2^{24}$ and $t = 32$ which will be good for $\frac{n}{2t} = 2^{18}$ signatures and secure against coalitions of $\frac{2^{12}}{2} = 2^{11}$ users.

The advantage of this scheme over the previous one is that there is a guarantee about what a group of colluders may know about the set of keys belonging to a user. The disadvantage is that there are only as many predefined private/public key pairs as there are blocks of the design, and there is a chance that two different individuals will end up with the same key pair. This means that, in this case, T must take similar precautions with regard to the selection of parameters and the distribution of private keys as discussed in section 2.3.

6 ID-based encryption from symmetric primitives

An ID-based public key encryption scheme is easily derived from the scheme described in subsection 4.2. As for ID-based signature schemes, in the naive approach the trusted centre generates n (public, private) key pairs where a private key S_i consists of a set $\mathcal{K}_i$ of secret keys and the public key $P_i = (\{e_K(X_i); K \in \mathcal{K}_i\}, X_i)$ for distinct identifiers X_i. The public key of user A with identifier i_A is $P_{f(i_A)}$. As for signature schemes, the trusted centre could use hash-tree methods to authenticate the public keys.

In an optimised approach to the generation of an ID-based encryption scheme, the trusted authority T would generate a collection of triples $(K_i, e_{K_i}(X_i), X_i)$, $i = 1, \ldots, N$. The authority T then publishes all the pairs $(e_{K_i}(X_i), X_i)$, $1 \leq i \leq N$.

A user determines the public key of user A with identifier i_A by applying the appropriate key-finding function to determine an n-subset of $\{1, \ldots, N\}$. The public key for user A consists of the collection of n values $e_{K_j}(X_j)$ for which $j \in f(i_A)$. The private key for user A then consists of the corresponding collection of values K_j.

Analogously to the discussion in section 5.2, an alternative optimised approach would involve generating subsets of keys for individual users using a block design instead of randomly. In this case, again as previously, the key-funding function f would map the user name to a block of the design.

An adversary who is able to subvert r of the keys held by user A would have probability $\frac{r}{n}$ of decrypting a ciphertext. Otherwise the adversary would be faced with the difficulty of breaking the block cipher.

7 Summary and conclusions

We have shown that any public key cryptosystem may be used to derive an identity-based public key cryptosystem. In particular those public key cryptosystems that are based on symmetric primitives may be so used. We have given constructions of identity-based public key cryptosytems based on symmetric primitives using this technique. Finally, we have shown that the efficiency of such systems may be improved over that of the naive approach by using both one-way functions and block designs.

References

[1] D. Bleichenbacher and U. Maurer. On the efficiency of one-time digital signatures. In K. Kim and T. Matsumoto, editors, *Advances in*

Cryptology — ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings, number 1163 in Lecture Notes in Computer Science, pages 145–158. Springer-Verlag, Berlin, 1996.

[2] R. Blom. Non-public key distribution. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *Advances in Cryptology: Crypto 82*, pages 231–236, Santa Barbara, Ca., 1983. Plenum Press, New York.

[3] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, number 2139 in Lecture Notes in Computer Science, pages 213–229. Springer-Verlag, Berlin, 2001.

[4] J. N. Bos and D. Chaum. Provably unforgeable signatures. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, Berlin, 1993.

[5] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer-Verlag, Berlin, 2001.

[6] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer-Verlag, Berlin, 2001.

[7] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, **IT-22**:644–654, 1976.

[8] S.-H. Heng and K. Kurosawa. k-resilient identity-based encryption in the standard model. In T. Okamoto, editor, *Topics in Cryptology — CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 67–80. Springer, Berlin-Heidelburg, 2004.

[9] J. W. P. Hirschfeld. *Projective geometries over finite fields.* Oxford University Press, Oxford, 1979.

[10] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770–3, Information technology—Security techniques—Key management; Part 3: Mechanisms using asymmetric techniques*, 1999.

[11] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 14888–2, Information technology — Security techniques — Digital signatures with appendix — Part 2: Identity-based mechanisms*, 1999.

[12] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 18033–3, Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*, 2005.

[13] L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International, Computer Science Laboratory, October 1979.

[14] W.-B. Lee and K.-C. Liao. Constructing identity-based cryptosystems for discrete logarithm based cryptosystems. *Journal of Network and Computer Applications*, 27:191–199, 2004.

[15] T. Matsumoto and H. Imai. On the key predistribution system: A practical solution to the key distribution problem. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, number 293 in Lecture Notes in Computer Science, pages 185–193. Springer-Verlag, Berlin, 1988.

[16] U. M. Maurer and Y. Yacobi. A non-interactive public-key distribution system. *Designs, Codes and Cryptography*, **9**:305–316, 1996.

[17] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

[18] R. C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, **21**:294–299, 1978.

[19] R. C. Merkle. A certified digital signature. In G. Brassard, editor, *Advances in Cryptology — Crypto '89*, number 435 in Lecture Notes in Computer Science, pages 218–238. Springer-Verlag, Berlin, 1990.

[20] C. J. Mitchell. Public key encryption using block ciphers. Technical Report RHUL-MA-2003-6, Mathematics Department, Royal Holloway, University of London, September 2003.

[21] A. Perrig. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM Conference on Computer and Communications Security, CCS 2001*, pages 28–37. ACM Press, 2001.

[22] J. Pieprzyk, H. Wang, and C. Xing. Multiple-time signature schemes secure against adaptive chosen message attacks. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15,*

2003, Revised Papers, volume 3006 of *Lecture Notes in Computer Science*, pages 88–100. Springer-Verlag, Berlin, 2004.

[23] M. O. Rabin. Digitalized signatures. In R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.

[24] L. Reyzin and M. Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In L. M. Batten and J. Seberry, editors, *Information Security and Privacy, 7th Australasian Conference, ACISP 2002, Melbourne, Australia, July 3-5, 2002, Proceedings*, volume 2384 of *Lecture Notes in Computer Science*, pages 144–153. Springer-Verlag, Berlin, 2002.

[25] Q. Tang and C. J. Mitchell. Cryptanalysis of a technique to transform discrete logarithm based cryptosystems into identity-based cryptosystems. Technical Report RHUL-MA-2005-4, Mathematics Department, Royal Holloway, University of London, March 2005.

[26] S. Tsujii and T. Itoh. An ID-based cryptosystem based on the discrete logarithm problem. *IEEE Journal on Selected Areas in Communications*, **7**:467–473, 1989.

[27] A. Weber. Secure communications over insecure channels (1974), by Ralph Merkle, with an interview from the year 1995. www.itas.fzk.de/mahp/weber/merkle.htm, January 2002.